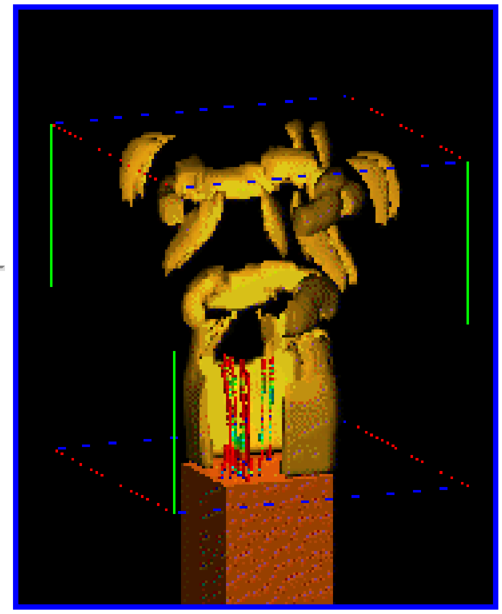
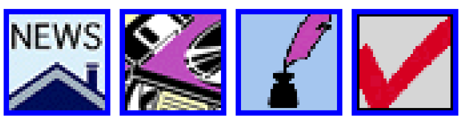
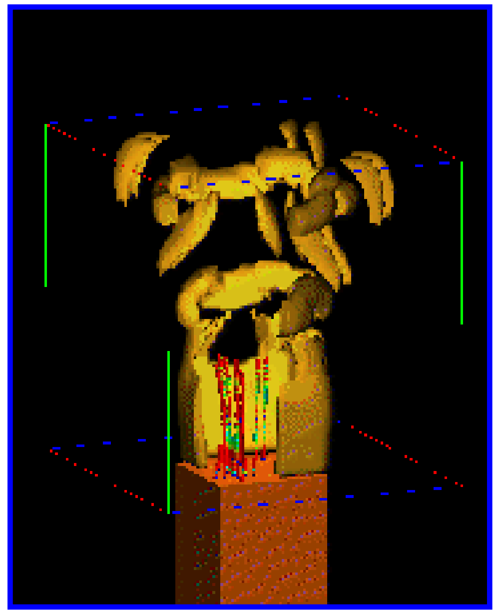
May - June 1996, Vol.2, No. 17





NAS Unveils Large-scale Interactive Visualization System



by Steve Bryson

The NAS Systems Division recently announced the Large-Scale Interactive Visualization Environment (LIVE), a system dedicated to the interactive visualization of very large datasets. LIVE will be available to eligible researchers from both within and outside of NAS. Potential users must first submit a proposal, outlined in the LIVE solicitation. Proposals are being accepted on an ongoing basis.

Developed by the NAS data analysis group, the LIVE system, located in the NAS Visualization Lab, consists of a Silicon Graphics Inc. Onyx system, with four R8000 processors, 5 gigabytes (GB) of physical memory, and two Reality Engine II graphics pipelines. The Onyx is connected to 200 GB of fast disk (with a tested read rate of 250 megabytes per second) and a variety of display and input options, including the newly acquired Interactive Workbench, built at Stanford University, and the Virtual Windtunnel BOOM apparatus.

Datasets in computational fluid dynamics (CFD) have grown dramatically over the years. Five years ago, gigabyte-sized unsteady datasets were considered "large" -- now, datasets in excess of 200 GB are not uncommon. Further, the detail and complexity of these simulations have increased, presenting CFD researchers with increasingly complicated phenomena. Visualizing large datasets and complex phenomena has been difficult with conventional visualization software and hardware.

Fast Response for Near-real-time Use

LIVE is oriented toward the visualization of these large, complex datasets. Two of the hardest aspects of scientific visualization are data management and data complexity. LIVE addresses the problem of data complexity by allowing "near-real-time" interactive visualization, which means that the result of a user's action is displayed in less than .5 seconds.

This fast response time allows users to continuously explore data, looking for interesting phenomena using interactive visualization techniques. The near-real-time technique is supported by the Virtual

Windtunnel, a NAS-developed software package.

Traditionally, such near-real-time interaction requires that all the data be in physical memory. The 5 GB of memory in the LIVE system may sound like a lot, but it barely addresses the problem of 200-GB datasets. To allow exploration of these larger datasets, a 200-GB, highly striped disk vault containing 96 individual disks, is included as part of the LIVE system. This disk vault should allow researchers to load and examine successive timesteps of data at near-real-time rates.

Software For Steady and Unsteady Flow

To perform scientific visualizations on LIVE, several NASA-developed software packages are available:

- **PLOT3D** and **FAST** for steady flow
- **UFAT**, the Virtual Windtunnel, and **elVis** for unsteady flow

Use of the system will be in one of two modes: batch for **UFAT**-type software packages and interactive for real-time user exploration of datasets using software like the Virtual Windtunnel.

Some users will require dedicated access to the LIVE system. For example, some problems may fill the 200-GB disk -- in fact, NAS is asking for such problems -- or, software such as the Virtual Windtunnel will require all CPUs for real-time response. Because it is expected that the majority of users will require dedicated access, time on the system will be scheduled in advance.

When using the LIVE system, users may explore a dataset, possibly video recording the exploration in real time. Users may also create high-quality scripted video in batch mode. Video and hardcopy output are produced using the capabilities of the NAS Visualization Lab (see *NAS News*, September-October '95).

Solicitation is Online

- Detailed description of the LIVE system
- Solicitation
- LIVE proposal form

You can also send an email request for information to **live@nas.nasa.gov**. Proposals are selected by the LIVE project team, which will also assist in the use of the system. Once accepted, users will receive scheduled time on the system.

Progress Continues on Supercomputing Consolidation Effort

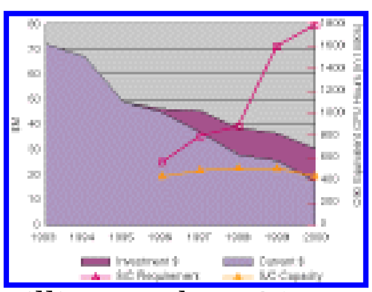
by Eric A. Hibbard

The NASA supercomputing consolidation effort reached a critical milestone with the submission of an implementation plan to NASA Headquarters in late January. This plan continued the analysis, begun last August, that preceded Ames Research Center's designation as the Agency's lead center for supercomputing management. (For background information, see the November-December '95 issue of *NAS News*.)

During this phase, NASA's supercomputer inventory was updated and each system was categorized as performing either production or research and development (R&D) supercomputing. Conceptually, the difference between the two categories is that production supercomputing is used as a tool to solve core Agency R&D science and engineering problems, whereas R&D supercomputing focuses on the supercomputers themselves (for example, new computing methodologies, assimilating new technologies, and increasing efficiency).

The rough rule of thumb used to differentiate between the two functions was that NAS- or HPCC-associated programs were categorized as R&D supercomputing -- and therefore exempt from consolidation at this time -- while everything else was considered production supercomputing -- and thus subject to the Agency's supercomputing consolidation.

Key Findings for Production Systems



Approximately $139 million is invested in NASA's production supercomputers, with current lease and maintenance obligations of about $77 million for these systems. Using FY93 as the baseline, current projected funding reflects a 39.6 percent -- or $228.8 million -- reduction in the overall funding of production supercomputing for the period FY94 through FY00 (see Figure 1, "Current $"). In FY00, NASA's production supercomputing budget will drop to about $22.5 million, or about 31 percent of the FY93 budget.

With currently identified funding levels, the Agency's annual supercomputing capacity is estimated to be in the range of about 450,000 CRAY C90 CPU hours (see Figure 1,"S/C Capacity"). To provide a common unit of reporting for supercomputing capabilities, all CPU metrics were normalized to equivalent CRAY C90 single-processor hours. However, the combined identified production supercomputing requirements for all NASA Enterprises (for example, Aeronautics, Earth Sciences, Space Sciences, Space Flight) range from about 790,000 hours in FY97 to 1,750,000 hours in FY00 (See Figure 1 "S/C Requirement").

If the current trend of increasing performance by about 44 percent annually for the same investment is factored in, it is projected that the Agency could meet its future production supercomputing needs with only a modest investment in both supercomputers and mass storage technology (See Figure 1 "Investment $").

Major Plan Recommendations

At press time, NASA Headquarters management was still reviewing the implementation plan and Ames was awaiting final approval. The plan submitted to Headquarters included the following major recommendations:

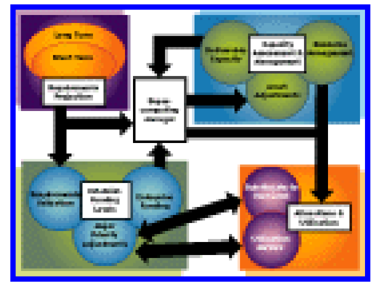
- Implement a management team to be lead by an Agency supercomputing manager and deputy located at Ames, who are advised by a Supercomputing Technical Advisory Board and a Supercomputing User Group. Policy and management oversight are to be provided by a Supercomputing Management Board of Directors, with members consisting of the associate administrators (or designees) from NASA Headquarters' Code M (Space Flight), Code R (Aeronautics, chair), Code S (Space Science), and Code Y (Earth Science), along with the center directors of the production supercomputing providers.

- After validating the projected requirements for each enterprise, resolve the projected budget shortfall for production super-computing. The enterprises have the option of prioritizing their requirements and funding a smaller set of requirements than that previously identified by the various program offices.

- Proceed immediately with consolidating the seven current NASA supercomputing providers into three centers (Ames, plus Marshall and Goddard Space Flight Centers) by October 1.

- Ensure that enterprise-specific barriers to realizing cross-utilization of Agency supercomputing assets are eliminated. Proceed immediately with bringing all production supercomputing resources into compliance with a common configuration (including consistent group and user identifications for all involved systems, similar versions of the operating systems, and standardization of software licenses).

  This means that, irrespective of the resource location, scientists and engineers will have easy access to the resource and/or facility that is most appropriate for solving a specific problem, independent of enterprise -- within the limits of their allocations.



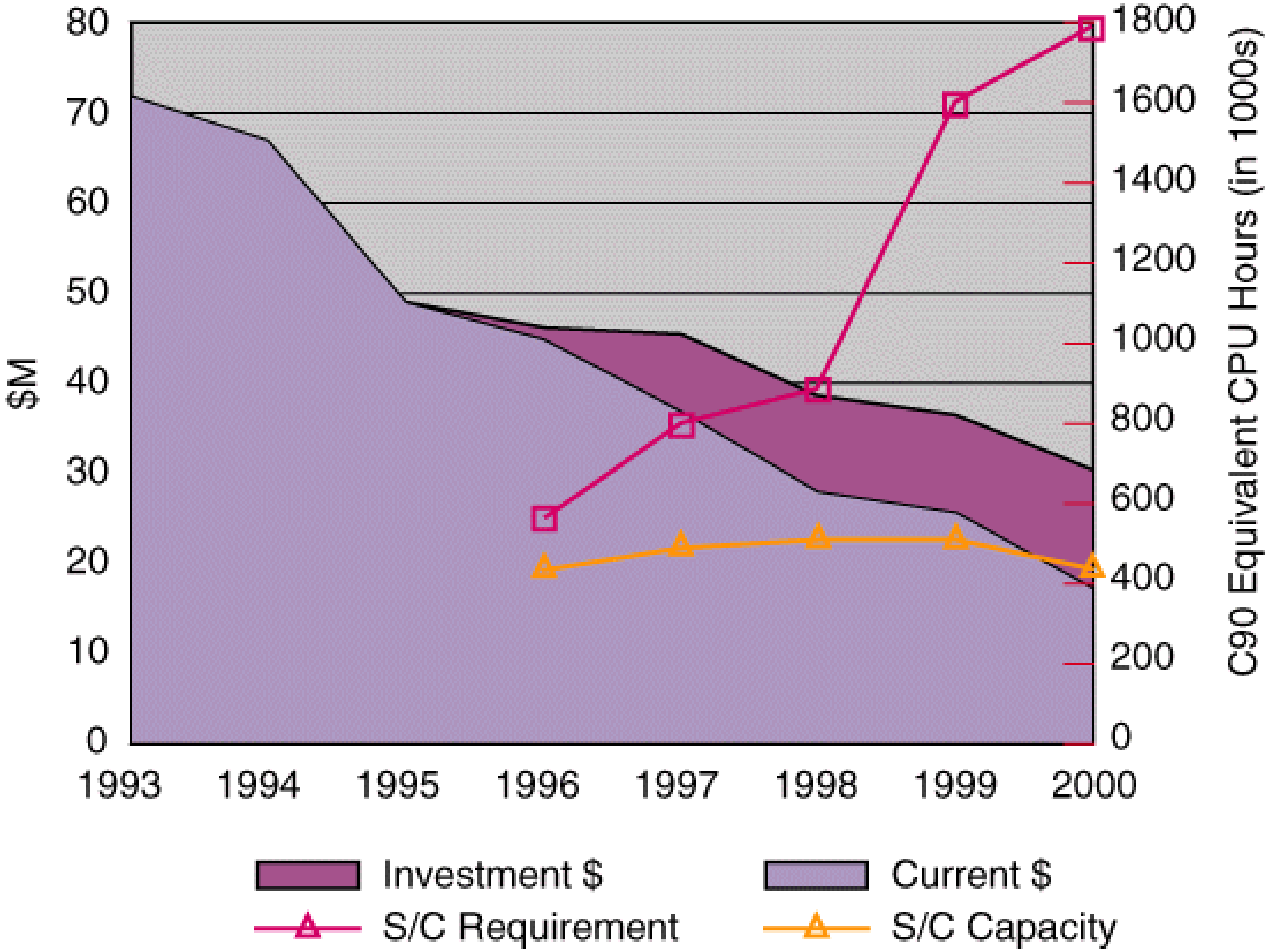
- Proceed with re-engineering the following processes for managing production supercomputing: requirements, assets, funding, and allocation (see Figure 2).

What's Next?

Some next steps in the consolidation process include:

- Define the roles and responsibilities for the lead center for supercomputing, including the identification of "expert centers."
- Implement processes to coordinate requirements gathering, budgeting and funding,
- capacity management, and allocations for the entire Agency.
- Determine production supercomputing requirements that could potentially be met through outsourcing. Solicit industry comments on a draft Request for Proposal to help assess the viability of this option.
- Adjust the current NASA production supercomputing assets to maximize available CPU cycles and minimize overall costs.
- Develop and integrate activities to implement a NASA supercomputing metacenter.

Next Article | Contents | Main Menu | NAS Home



$M
80
70
60
50
40
30
20
10
0
1800
1600
1400
1200
1000
800
600
400
200
0
C90 Equivalent CPU Hours (in 1000s)
1993
1994
1995
1996
1997
1998
1999
2000
Investment $
Current $
S/C Requirement
S/C Capacity

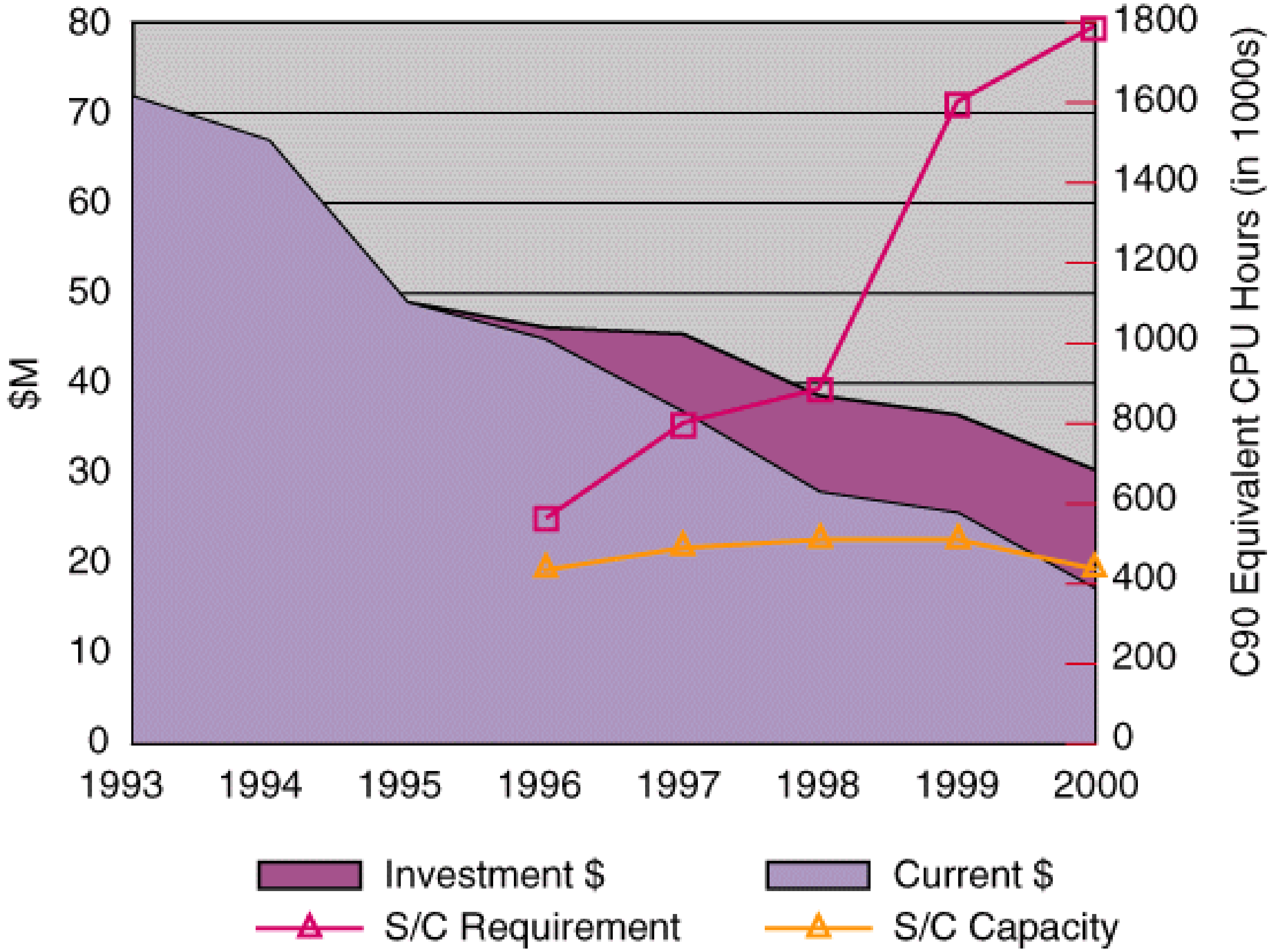


Figure 1. *NASA production supercomputing funding, capacity, and requirements.*

to the article.

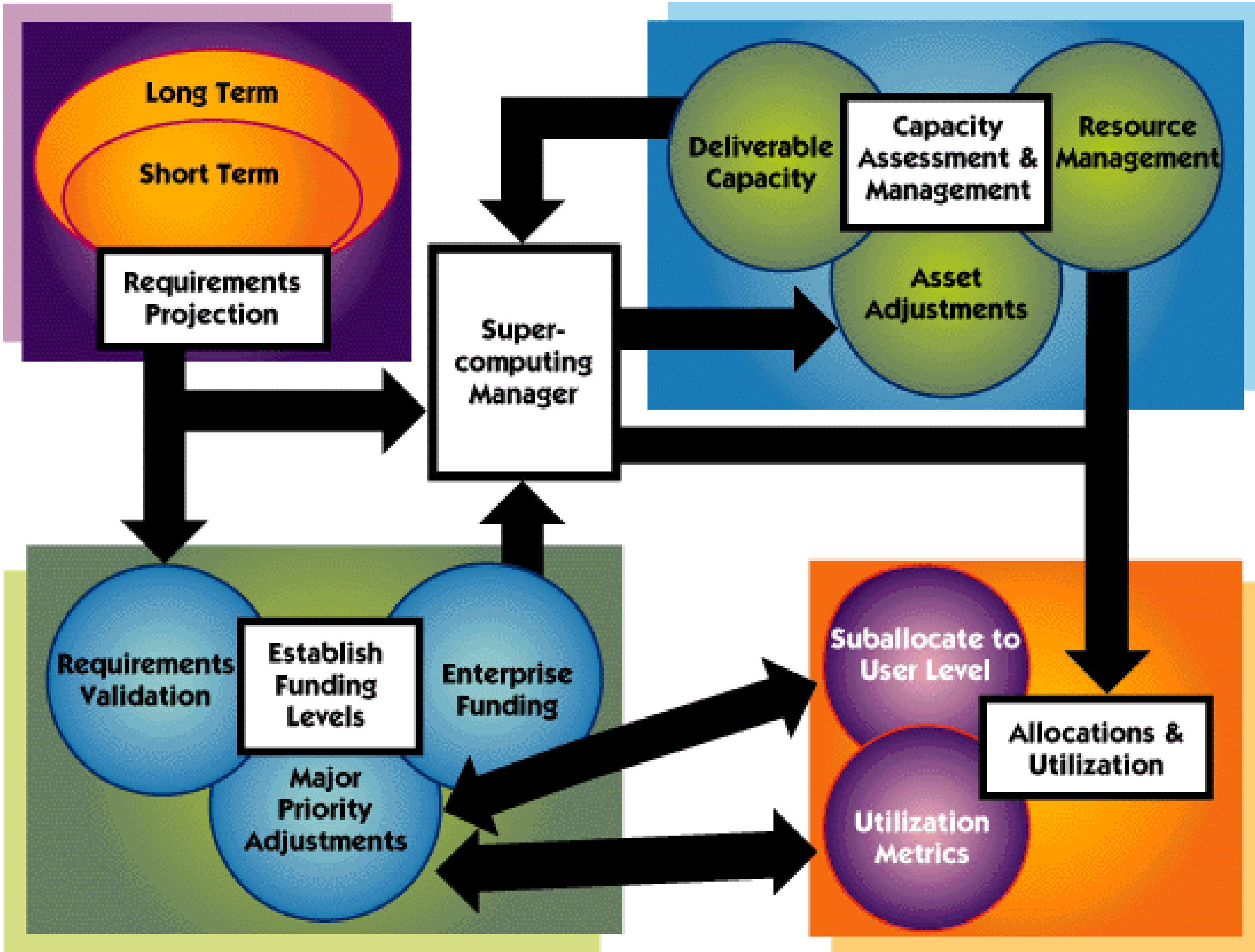


Figure 2. *NASA supercomputing management process.*

[to the article](#).

NAS Team Builds DARWIN Networks for `Faster, Better, Cheaper' Remote Site Aerotests

by Elisabeth Wechsler

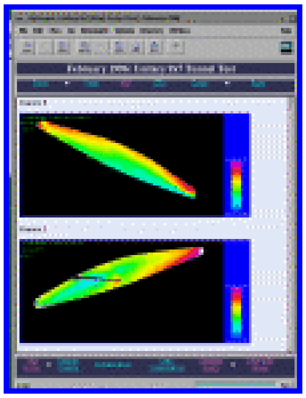
The NAS networks group recently completed a project that connects the AEROnet wide area network and DARWINnet, a secure local area network at Ames Research Center, to facilitate remote access to wind tunnel and other aircraft design tests. The work is part of the DARWIN Integrated Product Team (IPT). DARWIN, which stands for "Development Aeronautics Revolutionizing Wind tunnels and Intelligent systems for NASA," a joint Aeronautics and Information Systems effort, is chartered as a pathfinder for the new Development Aeronautics role at Ames. Dennis Koga is the overall lead.

Extending technology that was originally developed for two related wind tunnel projects, Integration of Numerical and Experimental Wind Tunnels (IofNEWT) and Remote Access Wind Tunnel (RAWT), the NAS team "provided the underlying infrastructure for software to link the Ames wind tunnels to the aeronautics community," explained Anthony Lisotta, NAS project lead. Not only will this save travel costs and personnel time, but test results can be made available more quickly and shared interactively, which shortens the design process and -- most important -- the critical time-to-market factor.

Flight Test at Dryden

The NAS team has worked since September '95 to implement the first DARWIN-supported flight test of an F-16XL aircraft (manufactured by General Dynamics, then modified by NASA), involving real-time data collection and analysis, at Dryden Flight Center in Southern California. The tests, a collaboration between several industry and NASA representatives, have focused on supersonic active laminar flow techniques.

In addition, the NAS team connected the Ames 12-ft. and 40 x 80 ft. wind tunnels to DARWINnet in support of industry tests. Since May 1, the NAS team has provided training and is turning over its part of the project to DARWIN and wind tunnel support staff.



The user interface for DARWIN is World Wide Web-driven and Web-accessed, explained David Korsmeyer, information systems manager for the DARWIN team. The collaborative software tools include Silicon Graphics Inc. (SGI) InPerson software -- which lets two or more people separated by thousands of miles interact with voice, white boards, and video via their workstations -- and Video Imaging Tool (VIC), which allows aircraft model viewing at remote test sites with no one physically present. Users can perform database queries during and after tests via the interactive data analysis tools.

These software tools, called the DARWIN Workspace Environment, are combined with hardware components -- SGI Indy workstation, rack, router, video switcher, color photography printer, and data encryption box -- to create the DARWIN remote access system.

Log Into Wind Tunnel Tests

Equipped with this system and a password, a future researcher, manager, or technician will be able to log into any NASA aerotest site -- wind tunnels at Ames, Langley, or Lewis Research Centers or flight tests at Dryden. For example, the appropriate password would give access to the latest ongoing test results, test plan, or set of completed test results in the DARWIN database for the Ames 12-ft. wind tunnel.

The DARWIN team has written Java applets (software applications) that are sent over the network and which then build graphs from the test information for the requesting customer. "As you go through the levels of data, you can do more evaluation. If you see something `interesting,' you'll want to compare results with actual CFD data, for example, to verify and further understand your results," Korsmeyer said.

Security issues for DARWIN are taken very seriously, according to Korsmeyer, as aircraft companies are particularly concerned about protecting proprietary test information. "You always want to know who's using the system," Korsmeyer said. The following security procedures are in place: isolate access to test sites, authenticate each participant with a password specific to each test, and encrypt data via both software and hardware methods.

The infrastructure allowing DARWIN to work includes AEROnet, which ensures the isolation aspect of network security, Korsmeyer said. DARWINnet provides dedicated fiber to link all wind tunnels, the NAS Facility, simulators, and miscellaneous test facilities at Ames.

NAS Group's Contribution

The NAS group's involvement with other DARWIN tests includes:

- DC-10 model in the 40 x 80 ft. wind tunnel, August '95. The IBM SP2 was used for data reduction for Microphone Array Phased Processing Systems (MAPPS) data, a computationally intensive analysis to determine areas of noise (indicating friction) on the aircraft. AEROnet was used to ship data to the customer. (For more information, see *NAS News*, September-October '95.)

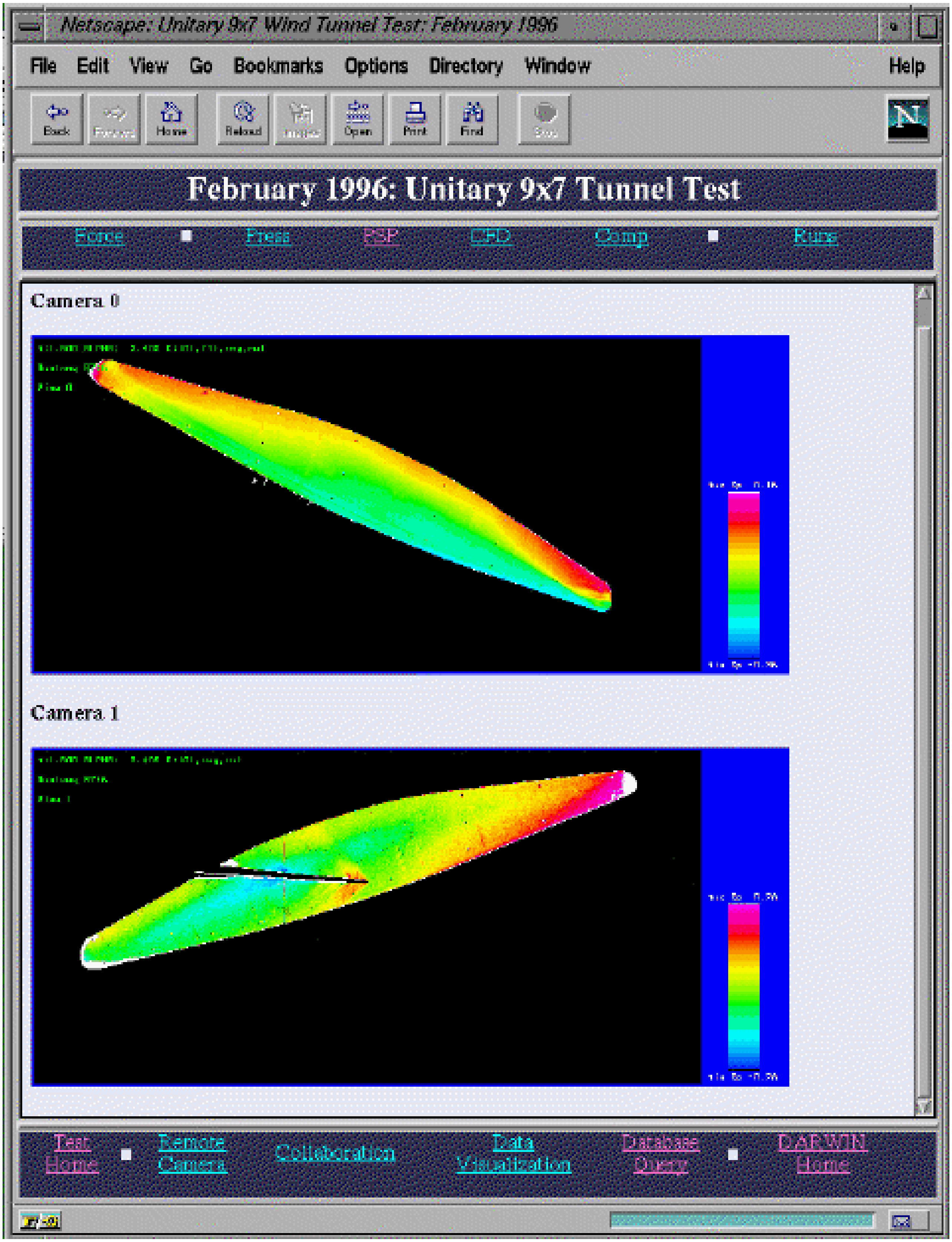
- The MAPPS instrument itself in the 40 x 80 ft. wind tunnel, scheduled for January '97. Results will probably be run on davinci, the SGI workstation cluster at NAS.

- 12-ft. pressurized wind tunnel -- first big test for the McDonnell Douglas high wing transport semi-span (half wing) model, scheduled for March '97, in which McDonnell Douglas and Langley representatives will receive real-time data via DARWIN.

- In addition, the DARWIN team has performed pseudo testing, reloading NASA-industry cooperative test data for simulations "to avoid issues of proprietary data," Korsmeyer said. An example, shown in the figure, is the Oblique All-Wing (airplane wing only) model data, an experimental high-speed transport concept, which involved several aeronautics companies.

Public information about the DARWIN project is available on the World Wide Web.

Next Article | Contents | Main Menu | NAS Home



This screen dump of the Netscape interface to the DARWIN online analytical tool shows upper and lower camer views of the Oblique All-Wing mode lin the Ames Unitary 9x7 ft. wind tunnel. Data available is shown as buttons on the data bar near top. A digital image of the data from pressure-sensitive paint (PSP) is captured by cameras mounted in the wind tunnel. The PSP is irratiated with ultraviolet light on the model to create the image, then a faux color mapping is done to distinguish pressure regions.

Red is the area of greatest pressure, followed by yellow and then blue, the area of lowest pressure. This image is available within minutes to remote site viewers.

to the article

Overset Grid Flow Solver Ready for Production Use

by Elisabeth Wechsler

The overset grid flow solver, OVERFLOW, developed by Ames scientists Pieter Buning, Dennis Jespersen and others, is being tested and evaluated using actual aeronautics industry configurations. Slightly different versions of the modular software tool, written in Fortran 77, are available for the CRAY C90, IBM SP2, and workstation cluster platforms by compiling the code differently.

OVERFLOW is designed to help CFD researchers and aircraft designers analyze whole vehicles more easily by simplifying the grid generation process. "In an overset grid scheme, grids are generated about individual components (such as a wing or fuselage), and then the overlapped grids are tied together to create a grid system about the complete vehicle," Buning explained.

Making It `Fast and Convenient'

Getting OVERFLOW ready for production means "making it fast and convenient to use on lots of different computing platforms," Buning said. Currently, the project involves collaboration with other NASA programs, such as the NASA Advanced Subsonic Technology (AST) Program High Lift Sub-element at Ames and Propulsion/Airframe Integration Sub-element at Langley Research Center, as well as the participation of technical staff at Boeing Commercial Aerospace Group and McDonnell Douglas Aerospace, who are also the primary customers. The project is funded by the NASA AST Program.

The project's industry partners are evaluating OVERFLOW and giving feedback on the accuracy and usefulness of the code, Buning said, adding that "industry interaction is most valuable for comparing results with experimental data." In addition, Boeing and McDonnell Douglas are giving input for usability enhancement through weekly phone conferences and occasional travel to corporate and NASA sites.

The NASA interaction "is most useful for pushing OVERFLOW's capabilities -- such as requiring more detailed simulations and developing multidisciplinary capabilities," he said.

The next major improvement in OVERFLOW was expected in April, and addresses multigrid acceleration. Since the flow solver is only part of the CFD process, ongoing work with OVERFLOW involves grid generation and post-processing.

Automatic Grid Generation

"We're trying to automate grid generation for subsonic transport configurations," Buning explained. In addition, post-processing takes the results of the flow solver and integrates the surface flow quantities to compute aerodynamic forces, such as lift and drag.

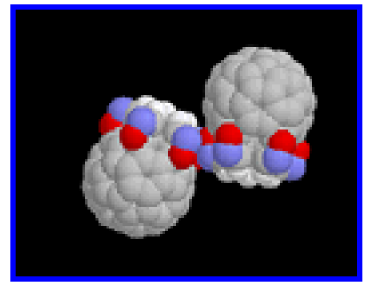
In June, a package for doing CFD automatically on wing/ fuselage/engine configurations will be available. After that, the program will focus on another package for high lift (flaps down) for the entire CFD process -- grid generation, flow solver, and post-processing, Buning said.

These packages are intended to help aeronautics companies do CFD automatically for a certain class of vehicle configurations, he noted, cautioning that the packages are not designed to handle helicopters or fighters, though the individual tools such as OVERFLOW have this capability.

Look for more information in an article by Buning in the July-August issue of *NAS News*.

[Next Article] [Table of Contents] [NAS News Home Page] [NAS Home Page]

Nanotechnology Offers Long-term Solutions for Space Technology, Chip Manufacturing



by Elisabeth Wechsler

Two nanotechnology-related workshops held at the NAS Facility in March underscored a new research and development thrust that is part of the government-wide High Performance Computing Research effort.

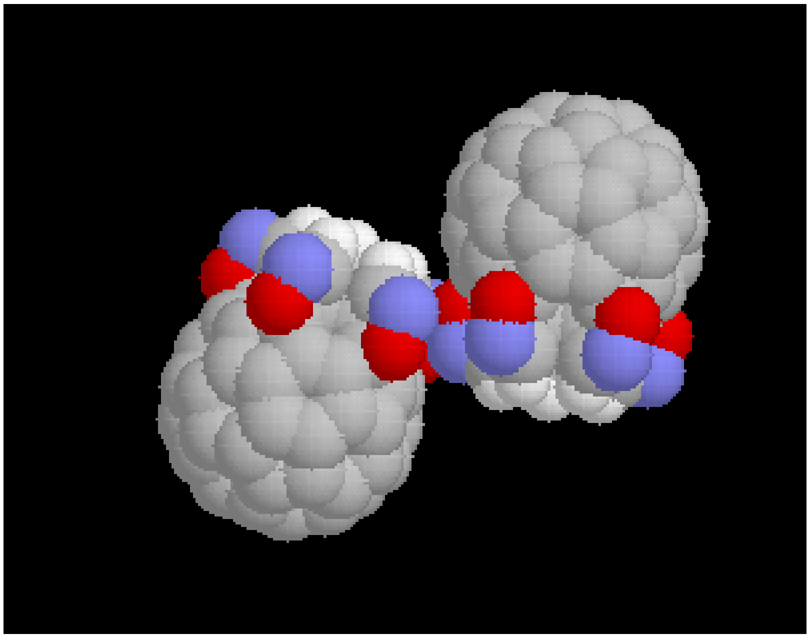
On March 4-5, the Computational Molecular Nanotechnology Workshop drew 75 participants from NASA centers, industry, and academia. The focus of this program was on progress achieved to date, as well as theoretical implications for producing atomically precise building blocks for a wide range of applications, including vastly superior launch vehicles and nano-robot designs.

More than a dozen presenters gave talks or held discussions on topics ranging from numerical techniques to molecular computer memory designs and self-assembly processes. Abstracts for the presentations are available.

The other workshop, Semiconductor Device Modeling, was held March 28-29. Some 110 participants, including significant representation from Silicon Valley computer manufacturers, as well Ames and academia, focused on nanoelectronics-related topics with medium-range application to silicon chip and circuit technology. Presentations included analyses of quantum effects, potential roadblocks to future progress, the outlook for photolithography, numerical techniques, technology computer-aided design (TCAD), and the conversion of large device simulation codes to highly parallel computer systems.

Proceedings from this workshop's 25 speakers are being published and will be available this month from NAS event coordinator Marcia Redmond. Send email requests to **redmond@nas.nasa.gov**; be sure to include your U.S. mail address for videotapes and/or proceedings. The latest NAS training information can be found on the World Wide Web.

The July-August issue of *NAS News* will devote a special section to presentations and discussions from these workshops and related research.



This miniaturized gear concept containing two molecules was developed by Al Globus at the NAS Facility in collaboration with William Goddard (Califorinia Institute of Technology) and Ralph Merkle (Xerox Palo Alto Reasearch Center). The hypotheses is that when one molecule is rotated, it will force the other molocule to rotate as well.

The long-term research goal is to design and build programmable molecular machines, with the effort representing on small step in the path. The collaborators all spoke at the Computational Molecular Nanotechnology Workshop, held at the NAS Facility, March 4-5.

Graphic created by Al Globus.

to the article

Options For Storing and Accessing Large Files on the Parallel Systems at NAS

by Steve Heistand

Once your code is running smoothly in parallel, the question arises as to what to do with all those output files. Large codes can easily produce many gigabytes of data, and storing these files is becoming a major problem for many NAS users. This article explains some options available for handling large amounts of data on the parallel systems at the NAS Facility.

The most convenient and straightforward approach would be to keep all files in the users' home filesystems on the parallel machines. But, with over 500 users on the IBM SP2 and almost 200 on the SGI POWER CHALLENGE cluster, the cost of required hardware would be very expensive.

Another solution is to have a large scratch filesystem on each system. This gives users a place to put their files that is easy to get to and has relatively fast access time. This solution has been implemented on both parallel systems at NAS, with a 15-gigabyte (GB) filesystem on the SP2 and a 9-GB filesystem on the cluster. These are both relatively new -- available since the beginning of the year -- and are heavily used. Current policy calls for files older than two days to be deleted, limiting the use of these filesystems to temporary storage only.

As both are NFS-mounted filesystems, the transfer rate is currently limited to about 3 megabytes (MB) per second. The SP2 will switch to PIOFS (IBM's Parallel I/O Filesystem) when PIOFS becomes more stable. (For more information, see "Using PIOFS Efficiently on the IBM SP2," *NAS News*, March-April '96.)

For long-term storage, the best place to keep files is on NAStore, the NAS mass storage system (two Convex 3820s named chuck and scott). Both machines have about 800 GB of file storage space (currently, 150 GB and 100 GB are available, respectively). These are archival systems, which means that most of the files are on tape and can be accessed within about 30 seconds, depending on how many people are using the system.

Both parallel systems are directly connected to NAStore via a high-speed HIPPI connection, which allows users to access files that are physically on the NAStore machines in codes that are running on the parallel machines.

Accessing Files From NAStore

There are two main ways to use files from the mass storage systems in your code: In the first, copy the files from chuck or scott immediately prior to running codes. This can be done either at the prompt prior to submitting a PBS job (not recommended) or within the PBS script right before running a program. To transfer the files, **ftp** from NAStore to node "b2003" on the SP2, or to davinci on the POWER CHALLENGE cluster.

To transfer files from inside a PBS script, add a line such as:

```
rcp scott:~/path_to_file_name/scratch1/username/file_name
```

Copying the file to a compute node that you have been assigned becomes rather tricky if using more than one node. If using only one node on either system, then try:

```
poe "rcp scott:~/file_name /tmp/file_name"
```

Reverse the procedure to move files back to NAStore at the end of the run, and remove any unnecessary files.

The second way of accessing files is to have the code move the files automatically. The following describes two locally written functions that will allow your code to get files from **chuck** or **scott**. The functions are:

```
Get_MSS_File (machine , remote_file_name , local_file_name)
Put_MSS_File(machine , local_file_name , remote_file_name)
```

Both of these functions return an integer value indicating success or failure. If this value is not zero, then an error occurred. These can be called from either Fortran or C with the same name and arguments. An example in C follows:

```
char *machine , *remote , *local;
machine = (char *)malloc(10*sizeof(char));
remote = (char *)malloc(100 * sizeof(char));
local = (char *)malloc(100 * sizeof(char));
machine = (char *)strcpy(machine,"scott");
remote = (char *)strcpy(remote,"~/sp2_data/input.dat");
local = (char *)strcpy(local,"/tmp/input.dat");
if ( Get_MSS_File(machine,remote,local) !=0 )
{/* an error has occurred in copy */ }
```

The above example will move a file from the sp2_data directory called input.dat to the local filesystem (/tmp) on whichever node this section of code is running.

In Fortran:

```
character*10 machine
character*100 remote
character*100 local
machine = `chuck'
remote = `~/sp2_data/output.dat'
local = `/tmp/solution.dat`
if ( Put_MSS_File(machine,local,remote)
  .ne. 0 ) then
    error has occurred.
endif
```

The previous example will move a generated solution file on the local node /tmp/solution.dat to an account on chuck.

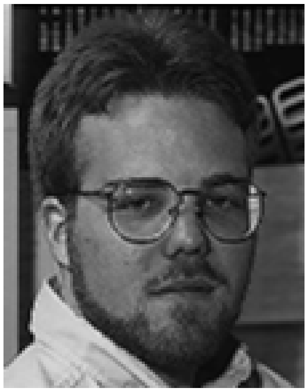
To use either of these routines, add the following to your program's link step; in Fortran, add the **-lMSS** flag.

In C code, add **-lMSS -lxlf90** (**xlf90>** is required, as part of the routine is written in Fortran). This library is located in the directory /usr/local/lib, so it may be necessary to add **-L/usr/local/lib**, if not already included in the link step of compilation.

A Note About Reliability

These routines will fail if the network to NAStore is down or extremely slow. Your code should be able to account for this in its error checking.

If you have questions or comments, contact the NAS scientific consulting group at **scicon@nas.nasa.gov** or call (415) 604-4444 or 1-800-331-8737.



Steve Heistand is a member of the NAS scientific consulting group. He joined the NAS Program in 1995.

Next Article | Contents | Main Menu | NAS Home

Converting Fortran Binary Files Between the CRAY C90 and Workstations

by Bob Hirsch and R.K. Owen

Fortran has two ways to output data -- either formatted or unformatted. Fortran formatted files are ASCII (or text) files that are readable and, therefore, portable from one architecture to another. With Fortran unformatted files, the data is written directly from the machine's internal representation to file and vice versa.

The advantages of Fortran unformatted files are that they tend to be more compact, far quicker for I/O, and suffer no loss of precision when read in by another Fortran program. For example, a Cray REAL variable is internally represented in 8 bytes -- the number of bytes written to a Fortran unformatted file for each REAL value. A typical formatted output requires about 20 bytes to accurately represent the same REAL value. Anything less leads to a loss of precision when the Fortran formatted file is read back in.

The formatting step, which translates the machine's internal representation to or from readable ASCII text, takes significant CPU time to perform. Fortran unformatted I/O requires no translation since the machine's internal representation is directly read from or written to data files. The chief disadvantage of Fortran unformatted files is the lack of portability from one architecture to another.

REAL	Cray	Vax/VMS	IEEE
length (bits)	64	32	32
exponent (bits)	15	8	8
mantissa (bits)	48	23	23
range (low)	3.67x10**-2466 2**-8190	2.93x10**-39 2**-127	1.175x10**-38 2**-128
range (high)	2.73x10**2465 2**8190	1.701x10**38 2**127	3.403x10**38 2**128
digits of accuracy	14	7	7

The tables highlight some differences between select common floating-point representations. (IEEE format is typically used for workstations and for the IBM SP2. There may be further differences between IEEE systems, especially with the way bytes are ordered in memory.) The most obvious difference is that Cray REAL is twice as large as the IEEE REAL.

Cray has several tools to aid in the conversion from these (and other disparate binary formats) to the Cray representation. The easiest to use is the Cray assign command. For the consummate programmer, there is a set of comparable Cray conversion routines. Two "home-grown" utilities, **wkconv** and **4Dconv**, can also help in some cases.

DOUBLE PRECISION	Cray	Vax/VMS	IEEE
length (bits)	128	64	64
exponent (bits)	15	8	11
mantissa (bits)	96	55	52
range (low)	3.67x10**-2466 2**-8189	2.93x10**-39 2**-128	2.23x10**-308 2**-1022
range (high)	2.73x10**2465 2**8190	1.701x10**38 2**127	1.80x10**308 2**1024
digits of accuracy	29	17	16

Cray assign command

The Cray assign command has several options, a few of which will be discussed here. For more information, see the man page **assign**(1) and the docview document **adviouser2**. The **assign** command is typically invoked from the command line, but can also be inserted into the program source code. The following is a command line example:

```
assign -F f77 -N ieee u:15
```

or equivalently placed in the source code with:

```
call assign(`assign -F f77 -N ieee u:15',ierr)
```

(Using this statement form rather than making an **ISHELL**() call is far more efficient and reduces the overhead of creating additional processes.)

The assign command has two useful options: **assign -V** displays any current assignments; **assign -R** removes all current assignments -- this is especially useful if there are "stale" assignments from a previous run that conflict with the current configuration.

The two most-used options for data conversion are **-F** for defining the Fortran record structure, and **-N** to specify the numeric format. These options tell assign to create an assignment file named ".assign" -- typically in the temporary $TMPDIR directory, with the appropriate I/O specifications. This file is parsed automatically by the Fortran executable; the executable tells the Flexible File I/O layer (FFIO) which sequences of interpretation to insert, if any. In a typical example of converting a Silicon Graphics Inc., Sun Microsystems Inc., or IBM RS6000 workstation Fortran unformatted data file, use the following to prepare the FFIO conversion layer:

```
assign -F f77 -N ieee u:25
```

where **u:25** is the logical unit number of the unformatted Fortran data file. It can also be specified as f:mydata.dat, if the source code opens logical unit number 25 with the filename, for example:

```
OPEN(25,file='mydata.dat',status='unknown').
```

For Digital Equipment Corp. (DEC) Ultrix workstations use **f77.vax** instead to account for the different byte ordering.

In the above example, for each Fortran record either read from or written to unit 25, the individual values are converted via the assigned FFIO layer. This allows the mixing of INTEGER, CHARACTER, REAL, or DOUBLE PRECISION data without any loss or confusion. As long as the data values fall within the representable range for that data type, then the conversion is completely transparent to the Fortran application or the user. The assign command does not translate the data -- it only specifies the FFIO layers to be used for the Fortran executable when performing I/O on the given logical units or files.

Avoiding Loss of Precision

Normally, when converting from Cray REAL to workstation REAL there is a loss of precision as the floating-point representation is truncated from 64 to 32 bits. Likewise, the reverse operation requires that the Cray floating-point representation be padded with extra zeros. Using **ieee_dp** instead of **ieee** in the numeric representation option **-N** allows Cray REAL (64 bits) to be converted to or from IEEE DOUBLE PRECISION (also 64 bits) with little loss of precision. Therefore, a portable application can be written in DOUBLE PRECISION for maximum precision on a workstation, and the same application can then be compiled with DOUBLE PRECISION turned off on the Cray using the **-dp** compiler option. The data files can be created or read on the workstation if the **assign** command -- as specified above -- is consistently used when running on a CRAY C90.

Special VAX/VMS Considerations

DEC VAX VMS users should generally use:

```
assign -F vms.s.tr -N vms u:35
```

for converting between VMS and Cray Fortran unformatted data files. The equivalent **vms_dp** converts between Cray 64-bit REAL and VMS 64-bit DOUBLE PRECISION. The real trick, however, is transferring the data file to or from a VMS machine without changing the VMS file structure. This is accomplished with the VMS **ATTRIBUTE** command. To prepare a binary file for transfer from a DEC VAX to a Cray, change the attribute to FIXED with:

```
ATTRIB/RTYPE=FIXED binary_file
```

before performing the binary ftp transfer to the Cray. To transfer a binary file from the Cray, do the following:

```
ATTRIB/RTYPE=VARIABLE binary_file
```

after the binary ftp transfer to the VAX.

Cray Conversion Routines

The standard Cray libraries contain a multitude of conversion routines. In fact, all the conversion layers accessible via the **assign** command have their counterparts in a library routine. For example, IEG2CRAY/CRAY2IEG provides the conversion capability between IEEE to or from Cray format for all data types -- including the special conversion provided by the **ieee_dp** FFIO layer mentioned above. However, it's the programmer's responsibility to read or write the data file in the correct Fortran record format. For more information, see the man page **intro_conversion(3F)**, which lists all available conversion routines.

Cray-to-workstation Conversion Utilities

To convert existing Cray Fortran unformatted data files that would be impossible or too costly to regenerate in IEEE format, there are two utilities: 4Dconv and wkconv. The two programs are invoked in basically the same way:

- **wkconv** ***input-file output-file***
- **4Dconv** ***input-file output-file***

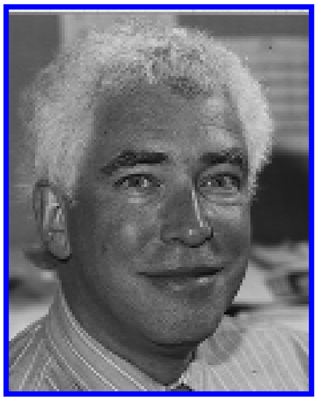
Both utilities expect as input a Cray (64 bits per word) Fortran unformatted file consisting of integers and single-precision real numbers, plus embedded record length information. 4Dconv produces an IEEE (32 bits per word) Fortran unformatted file, also with embedded record lengths, while **wkconv** produces an IEEE pure binary file, with data values only. So, use **4Dconv** to convert a file to be read by a Fortran language application and use wkconv for a file to be read by a C language application.

There are two general limitations to these conversion routines. The first is common to all conversion efforts: some very large or very small values that can be represented in the Cray floating-point format cannot be represented in IEEE format. Data of this nature need to be scaled in a way that brings them into the range of representable values.

The second limitation is that these two utilities expect only INTEGER and REAL data. No CHARACTER or DOUBLE PRECISION data types are allowed -- or these values will become garbled and useless. Existing data files containing any of these non-permissible values can be converted with a small user-written Fortran program that reads the Cray format input file, record by record, and does nothing but write the same records out in IEEE format with the help of the Cray assign command on the output file, as described above.

Using these Cray options for handling Fortran unformatted files instead of using formatted ASCII files -- especially for storing data that's rarely viewed -- may result in faster I/O and smaller files.

For further help, contact NAS User Services at (415) 604-4444 or (1-800) 331-8737 and ask to speak to a consultant or send email to **nashelp@nas.nasa.gov**. The NAS scientific consulting World Wide Web page also has a variety of user information of interest to users.



Bob Hirsch specializes in visualization tool support. He holds a bachelor's degree in psychology from Stanford University.

R. K. Owen is responsible for maintaining the third-party math libraries available to NAS users. He received a Ph.D. in physics from the University of California, Berkeley.

Both authors are members of the NAS scientific consulting team.

Next Article | Contents | Main Menu
NAS Home

REAL	Cray	Vax/VMS	IEEE
length (bits)	64	32	32
exponent (bits)	15	8	8
mantissa (bits)	48	23	23
range (low)	3.67x10**–2466 2**–8190	2.93x10**–39 2**–127	1.175x10**–38 2**–128
range (high)	2.73x10**2465 2**8190	1.701x10**38 2**127	3.403x10**38 2**128
digits of accuracy	14	7	7

Table 1 *Floating point representations*

to the article.

DOUBLE PRECISION	Cray	Vax/VMS	IEEE
length (bits)	128	64	64
exponent (bits)	15	8	11
mantissa (bits)	96	55	52
range (low)	3.67x10**–2466 2**–8189	2.93x10**–39 2**–128	2.23x10**–308 2**–1002
range (high)	2.73x10**2465 2**8190	1.701x10**38 2**127	1.80x10**308 2**1024
digits of accuracy	29	17	16

Table 2 *DOUBLE PRECISION representations*

to the article.

Using MPI-IO to Access Multidimensional Distributed Arrays

The March-April issue of NAS News reported on NAS's role in defining MPI-IO, an emerging standard for Parallel I/O. Members of the NAS parallel systems group are actively participating in the development of this standard. This article describes how MPI-IO can be used to write entire distributed arrays in a single collective write operation.

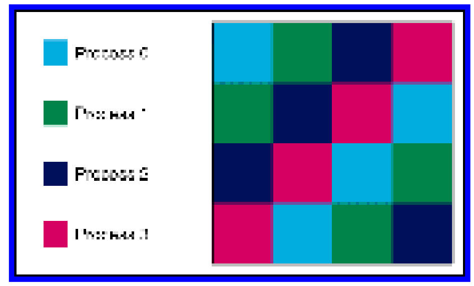
by Samuel A. Fineberg

Most scientific applications that run at NAS operate on large multidimensional (usually 3D) arrays that are distributed among a set of processes. These applications must read the arrays to obtain initial conditions for a calculation and write the arrays for later analysis. Ideally, these arrays should be stored on disk in flat files (for example, in PLOT3D format) so that they can be read by UNIX workstations or by other parallel applications that may or may not be using the same number of processes. Unfortunately, reading or writing such files has inherent problems. Users typically solve these problems by writing individual files from each process and using a postprocessor program to put them back together after a program has completed. This has drawbacks because it makes an extra step necessary after running a code, and because the number of files users must manage can be very large for massively parallel processor systems like the IBM SP2 -- which may have hundreds of processors. MPI-IO is designed to solve this problem by using collective I/O.

Collective I/O supports global operations (reading or writing an entire array, stored across a group of processes, in a single operation). By having processes work together, the NAS-developed collective buffering algorithm can be used to ensure that the actual file system accesses are of a size that is optimal for the underlying file system. Making this possible, however, requires mechanisms more powerful than the normal read and write commands because users must be able to describe how an entire array is laid out on disk, even though it may not be stored contiguously.

In MPI-IO, this mechanism is known as a "filetype." The filetype is an MPI derived datatype that describes the layout of the file as seen from a process. Each process can then open a file with a different filetype describing its portion of the file.

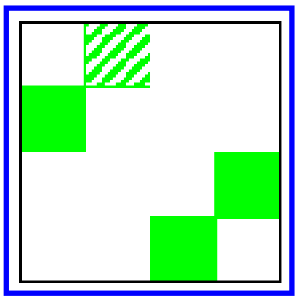
Creating a Filetype



For example, consider a 2D array that is distributed using the multipartition method (a simplified case of the 3D multipartition method used in the NAS BT 2.0 benchmark code), as shown in Figure 1. In this example, the 2D array is stored in smaller 2D "cells" across four processors. This could appear within each processor as follows:

```
     double precision array(dim/4, dim/4, 4)
```

where each small square in Figure 1 is a dim/4 array, and four of those are stored in each processor.



The first step in writing this to disk using MPI-IO is to create a filetype. The filetype describes where in the file a processor needs to write. For example, Process 1 of Figure 1 will have the filetype shown in Figure 2. The filetype is represented in MPI-IO as an MPI-derived datatype.

To assist in the creation of this filetype, MPI-IO has several "filetype constructors." These constructors will create filetypes for scatter/gather operations, High Performance Fortran-style array distributions, and general arrays. Other distributions can be created using the many datatype construction operations in the MPI standard. So, to create the filetype shown in Figure 2, use MPI-IO's subarray type constructor. The subarray constructor will create an MPI datatype describing any arbitrary subarray of a larger array.

Since the filetype in Figure 2 consists of four distinct subarrays, a subarray for each of the blocks must be created and joined into a single filetype using the operation MPI_Type_struct. The code to create the first of the subarrays -- the one shown with slashes in Figure 2 -- is as follows:

```
c sizes is the size of the overall
c distributed array
     sizes(0) = dim
     sizes(1) = dim
c subsizes is the size of the portion
c of the array we are describing
      subsizes(0) = dim/4
      subsizes(2) = dim/4
c starts is the array index of the
c upper left hand corner of the subarray
c (the 0,0 element) in the overall
c distributed array
      starts(0) = 0
      starts(1) = dim/4
c Other parameters indicate storage
```

```
c order (C or FORTRAN order) and the
c datatype of each array element
      MPIO_TYPE_SUBARRY(2, sizes, subsizes,
     $     starts, MPIO_FORTRAN_ORDER,
     $     MPI_DOUBLE_PRECISION, celltype(1),
     $     ierr)
```

The next step is to create similar subarrays for the four other subarrays, storing them in newtype(2), newtype(3), and newtype(4). To form this into a single filetype, use MPI_Type_struct:

```
c All block lengths are 1 (i.e., one
c subarray type)
      block_length(1) = 1
      block_length(2) = 1
      block_length(3) = 1
      block_length(4) = 1
c All block displacements are 0 (subarray
c types start at location 0 due to embedded
c holes)
      block_displacement(1) = 0
      block_displacement(2) = 0
      block_displacement(3) = 0
      block_displacement(4) = 0
      call MPI_TYPE_STRUCT(4, block_length,
     $     block_displacement, celltype,
     $     filetype, ierr)
     $     filetype, MPIO_OFFSET_RELATIVE, hints,
     $     fh, ierr)
```

Once a file has been opened with this filetype, data will only be written to the portions of the file described by the filetype. Therefore, simply write out the entire distributed array with the following collective write command:

```
      offset=0
      call MPIO_WRITE_ALL(fh, offset, array,
     $     1, status, ierr)
```

Only MPI-IO Supports Complex Data Layouts

The previous example shows that it is possible to create filetypes for complicated array distributions, and then write data in a single collective operation. Such functionality has previously only been available for parallel languages such as CM-Fortran or for regular data layouts. MPI-IO

allows you to distribute your data to optimize computational performance, without having to consider how this layout will affect I/O. MPI-IO can describe virtually any data layout, and works with any message passing program written in MPI.

Beware of Code Changes

Because MPI-IO is still being defined, some of the code shown here may not be current at the time of publication. However, the functionality provided by MPI-IO is not expected to change.

For more information, see the NAS MPI-IO page.

In the next issue of *NAS News*, Parkson Wong will present performance results of the NAS portable MPI-I0 library, PMPIO.

Samuel A. Fineberg works for MRJ Inc. in the NAS parallel systems group. He, along with Bill Nitzberg and Parkson Wong, is developing PMPIO.

Next Article Contents Main Menu
NAS Home

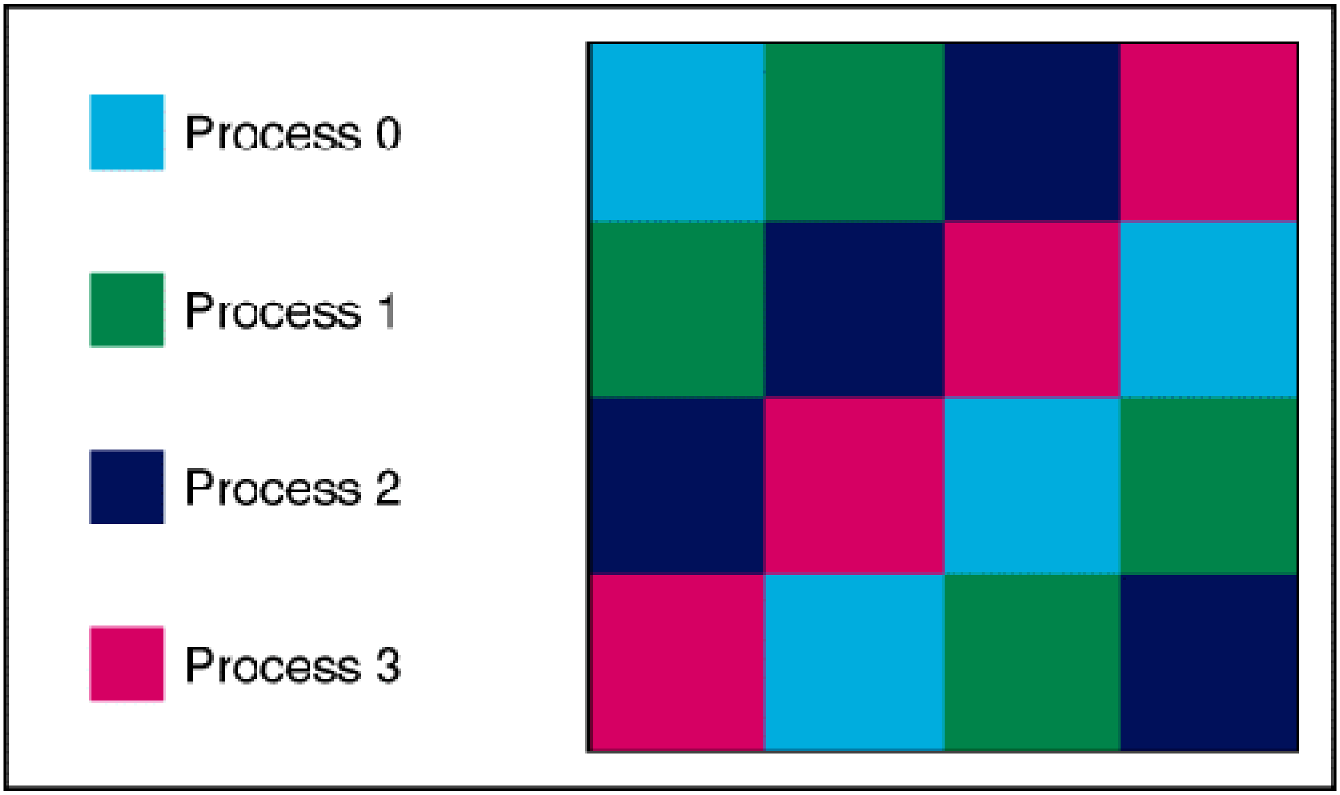


***Figure 1** An array distributed in multipartition fashion across four processors.*

to the article.

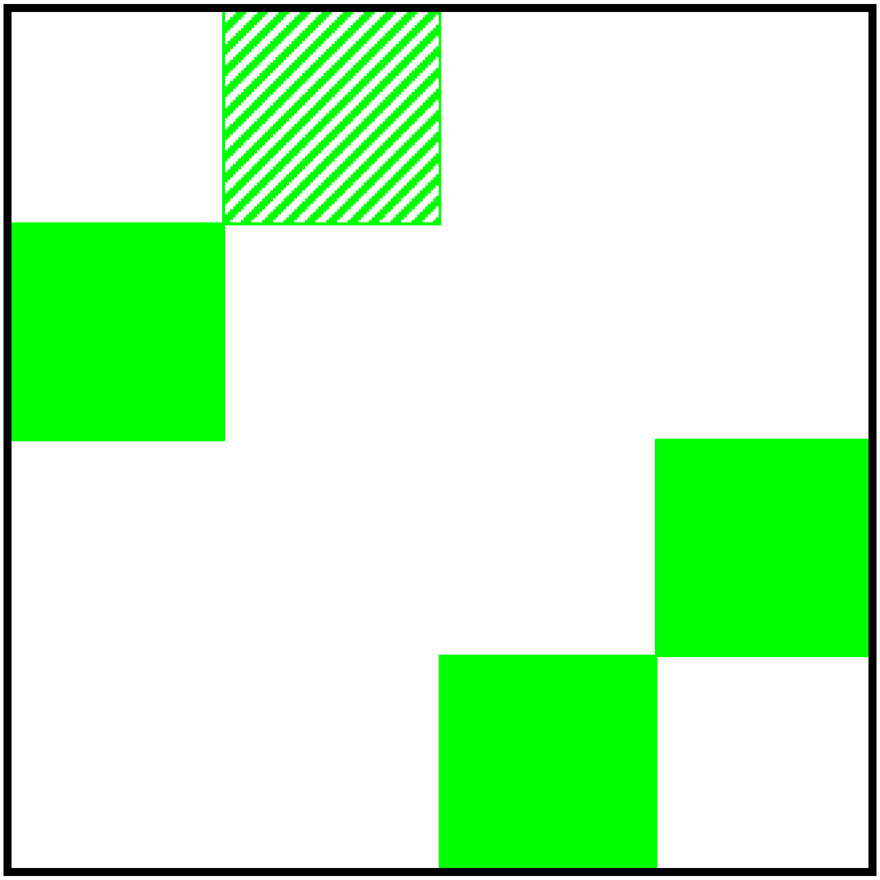


Figure 2 *Filetype example from process 1, using MPI-IO.*

to the article.

May-June 1996, Vol.2, No. 17

Executive Editor: Marisa Chancellor

Editor: Jill Dunbar

Senior Writer: Elisabeth Wechsler

Contributing Writers: Steve Bryson, Samuel A. Fineberg, Steve Heistand, Eric A. Hibbard, Robert Hirsch, R. K. Owen

Image Enhancements: Chris Gong

Other Contributors: Dan Asimov, Cristy Brickell, Pieter Buning, Glenn Deardorff, Dan DePauk, James Donald, Jude George, Al Globus, Bob Hood, Dennis Koga, David Korsmeyer, Anthony Lisotta, George Myers, Terry Nelson, Bill Nitzberg, Marcia Redmond, Mark Tangney, Dani Thompson, Tom Woodrow

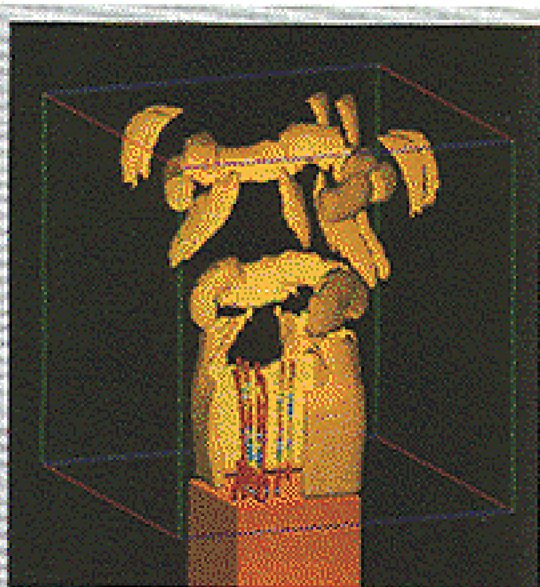
NAS NEWS

Volume 9 — May–June 1996

Progress Continues on Supercomputing Consolidation Effort

Continued on page 2

THIS ISSUE



NAS Unveils Large-scale Interactive Visualization System

Continued on page 2